Remarks

Claims 5 and 7, indicated as being allowable if rewritten in independent form (Action of July 28, 2004, page 2, ¶ 2), have been rewritten in such form. Claim 8, which depends of claim 7, and claims 9-10, which depend on claim 5, are believed now to be similarly allowable.

New claims 15 and 16 are similar to existing claim 14, but depend on claim 5 and 7, respectively, rather than on claim 1. These new claims thus present no new issues and are believed to be similarly allowable.

Claims 1-4, 6, and 11-14 again stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Deianov et al. 6,529,985 ("Deianov") in view of Hammond 6,463,583, either alone (page 2, ¶ 4) or in conjunction with other art (page 5, ¶ 12). These rejections are again respectfully traversed.

Claim 1, upon which the remaining claims under rejection depend, is directed to a system for intercepting API calls in a virtual memory environment. The system comprises an activation module (AM) and an interception module (IM) (Fig. 3). The interception module selectively provides modified functionality for the intercepted API calls. The activation module loads the interception module to occupy a location in a shared region of virtual memory as long as interception of the API calls is required. The activation module redirects the API calls by creating an alias to any page containing an entry point for an API call to be intercepted and writes the address of the interception module to the alias. The activation module provides to any instances of the interception module the original entry points for the API calls.

Deianov discloses a system for selective interception of API calls. It is an example of the prior art system (with its attendant disadvantages) described in the background portion of applicant's specification at page 3, lines 24-30, in which addresses in an interrupt vector table are altered to point to alternative code at the altered address. As shown in Fig. 1 of the patent, one embodiment of the system includes an interception module 111 and an interrupt vector table 113 in operating system (OS) address space 105, together with a modified loader program 121 and a system call

wrapper 125 in user address space 103. (In other embodiments, the system call wrapper 125 may be in OS address space.) In the interrupt vector table 113, selected pointers 114 to system calls 115 are replaced with pointers 118 to the interception module 111.

As the Examiner concedes, Deianov does not teach loading the interception module to occupy a location in a shared region of virtual memory as long as interception of the API calls is required, as claimed by applicant (page 3, ¶ 6). The Examiner asserts, however, that Hammond teaches this feature and that it would have been obvious to combine the two references because Hammond's teaching of loading the interception module "would improve the flexibility of Deianov's system by dynamically injecting the execution logic into a shared memory space of a window[ed] operating system" (pages 3-4, ¶ 7). Applicant respectfully disagrees.

Contrary to the Examiner's assertion, Hammond does not teach loading an interception module to occupy a location in a shared region of virtual memory as long as interception of the API calls is required, as claimed by applicant. Rather, Hammond discloses a system for dynamically injecting execution logic in the form of a dynamic link library (DLL) into a shared memory space of a windowed operating system (col. 2, lines 56-58; col. 8, line 36 to col. 9, line 3). Other than the fact that it loads an executable module in a shared user address space (which is certainly not new), Hammond has nothing to do with either Deianov or applicant's claimed invention. Certainly it would not suggest loading Deianov's interception module 111 into such shared user space when that reference so repeatedly teaches that it be loaded into the OS address space 105 (e.g., col. 5, line 64 to col. 6, line 2; col. 10, lines 26-30; col. 11, lines 54-58).

In responding to the above argument as previously presented, the Examiner contends that Deianov teaches that the interception module is loaded into the user address space and could be further loaded into the OS address space in a different embodiment (page 8, ¶ 22). The Examiner contends further that Hammond teaches loading an interception module into a shared region of virtual memory and that the location is "available to all processes (i.e. the interception is in the shared virtual memory as long as other processes still need it"). Therefore, the Examiner concludes, the combination of the two references teaches the claimed limitation. These statements simply miss the mark.

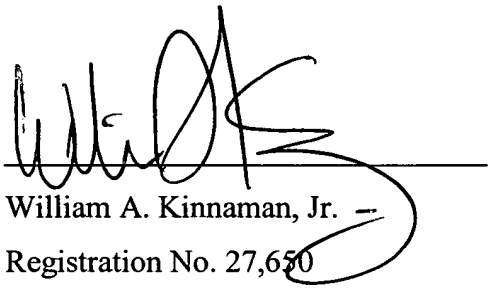
Regarding Deianov, the Examiner equates applicants' claimed interception module with both the initialization module 123 and system call wrapper 125 in user address space 103 rather than the interception module 111 in OS address space 105. However, while the system call wrapper 125 may contain code that is executed for intercepted system calls, it is not the place where execution starts for such intercepted system calls. Rather, such execution starts in the interception module 111 in OS address space 105. This is as a result of the pointers 118 in interrupt vector table 113 that have been rewritten to point to the interception module 111 rather than system calls 115. Thus, if the interrupt vector table 113 is considered to be the "alias" of applicant's claim (something applicant does not concede), the "interception module" whose address is written to that alias is the interception module 111, not the initialization module 123 or system call wrapper 125. That interception module 111, as applicant has previously noted, is always located in OS address space 105, where it cannot be modified on the fly as it could in user address space 103.

Regarding Hammond, the Examiner simply asserts that the dynamically injected execution logic of that reference is an "interception module" without offering any evidence to back this up. And even if Hammond's dynamically injected logic were considered to be an interception module, this still would not overcome the deficiencies of Deianov as a primary reference as noted above.

Conclusions

For the foregoing reasons, claim 1 and the claims dependent thereon are believed to distinguish patentably over the art cited by the Examiner. Entry of this amendment and reconsideration of the application as amended are respectfully requested. It is hoped that upon such consideration, the Examiner will hold all claims allowable and pass the case to issue at an early date. Such action is earnestly solicited.

Respectfully submitted,
RICHARD JOHN MOORE

By 

William A. Kinnaman, Jr.
Registration No. 27,650
Phone: (845) 433-1175
Fax: (845) 432-9601

WAK/wak